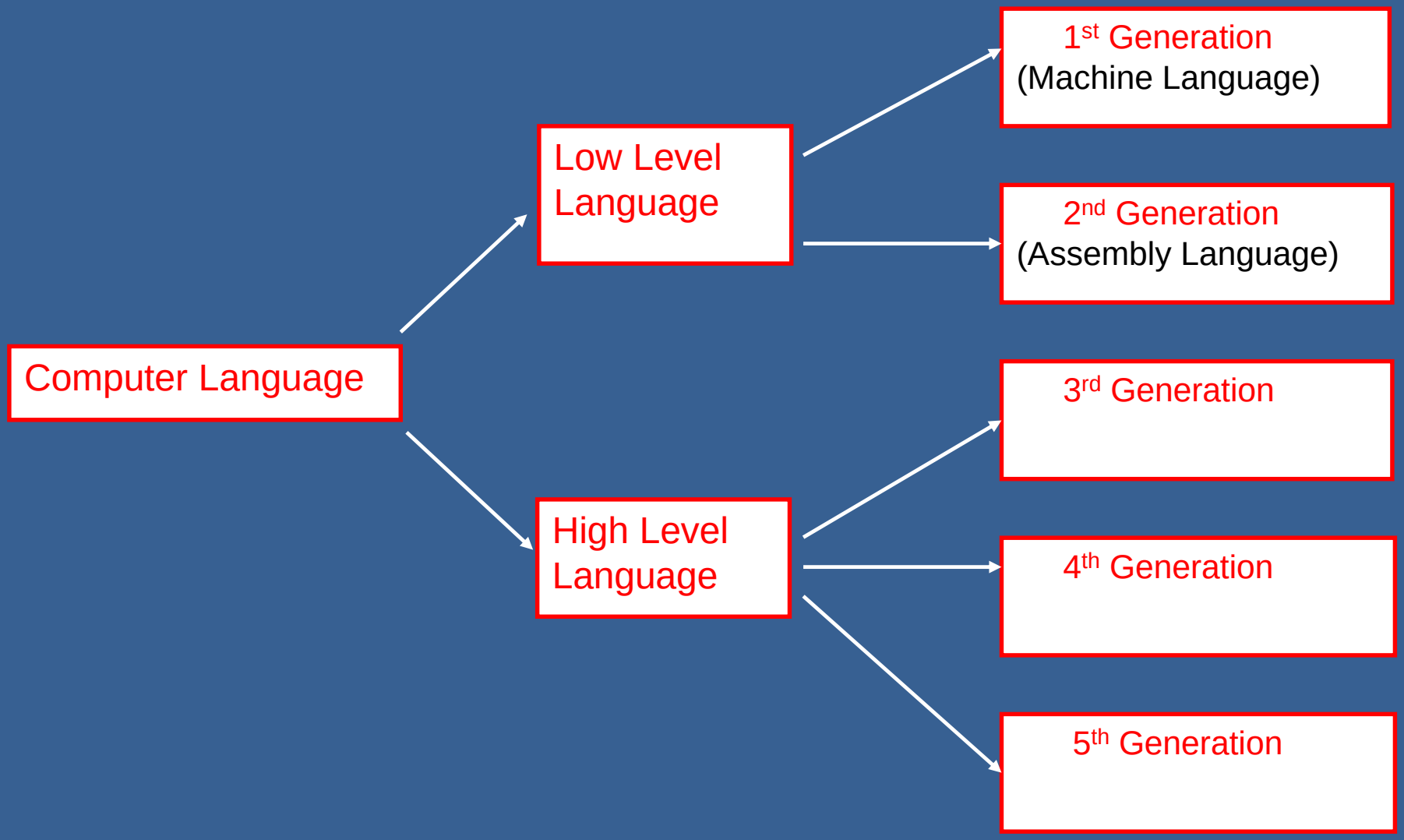
Programming Languages

BASIC TERMS

- A computer language is a set of symbols and rules used in constructing programs called syntax.
- Algorithm :- Step by step description that the program must perform arithmetical and logical expression to arrive at the solution.
- Flowchart:- A popular logic tool used for showing an algorithm in graphics form.

CLASSIFICATION



LOW LEVEL LANGUAGES

- Machine dependant.
- Close to the hardware
- Should have hardware knowledge to write a program

Examples :

- Machine Language
- Assembly Language

HIGH LEVEL LANGUAGES

- Machine independent.
- Easy to write and modify.
- Does not need knowledge of hardware.
- Productivity is high.
- Consume less time to write programs
- 5GL allow user-friendly facilities

PROGRAMMING LANGUAGE PARADIGMS

Procedural:

procedures, sequential execution of code are basic building blocks of program

- FORTRAN (FORmula TRANslating; John Backus, IBM, 1950s)
- ALGOL (ALGOrithmic Language, 1958)
- COBOL (COmmon Business Oriented Language, 1960)
- BASIC (Beginner's All-purpose Symbolic Instruction Code, John Kemeny and Thomas Kurtz, Dartmouth, 1964)
- Pascal (Niklaus Wirth, 1970)
- C (Dennis Ritchie, Bell Labs, 1972)

PROGRAMMING LANGUAGE PARADIGMS

Object-Oriented:

Program is designed around the *objects* required to solve the problem

- Smalltalk (Alan Kay, Xerox PARC, 1971)
- Ada (US Dept of Defense, 1975)
- C++ (Bjarne Stroustrup, Bell Labs, 1983)
- Java (James Gosling, Sun Microsystems, 1995)
- C# (Microsoft, 2000)

PROGRAMMING LANGUAGE PARADIGMS

Non-procedural languages

- HTML

  Hyper Text Markup Language
- JSP

  Java Server Page
- ASP

  Active Server Page
- SQL

  Structured Query Language

TRANSLATORS

Translator is used to convert source code into object code.

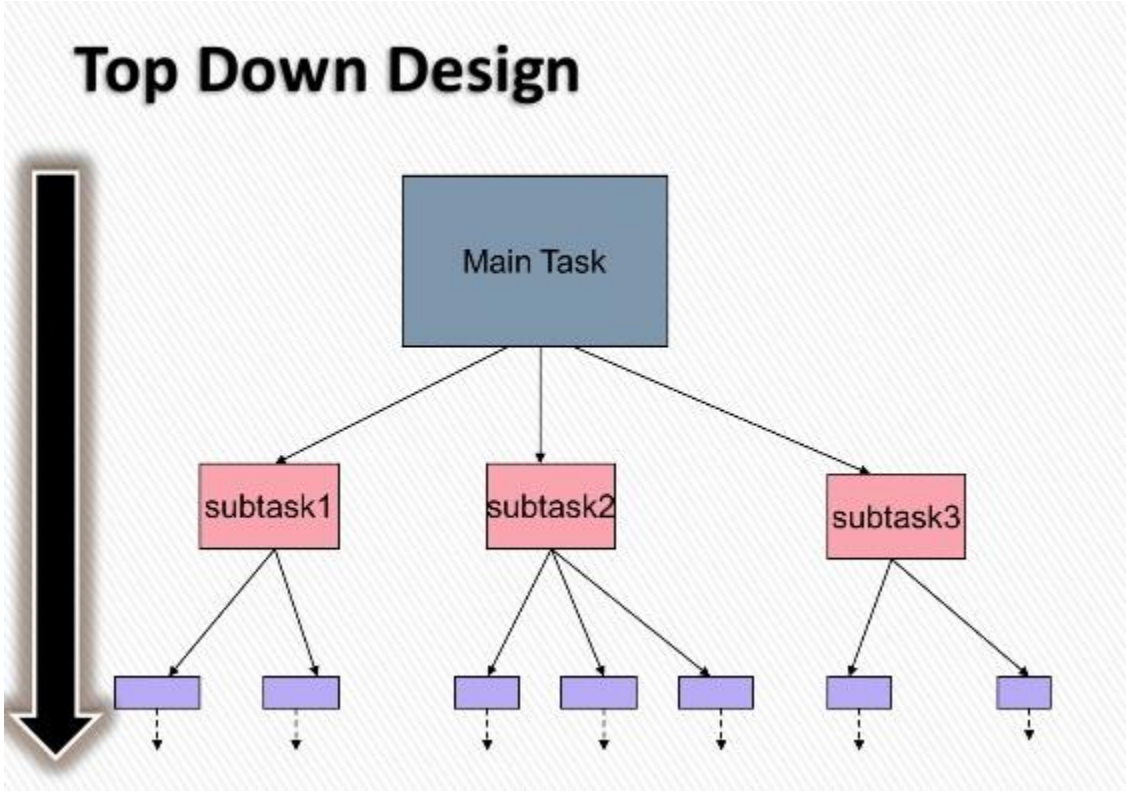
These are of three types

- Assembler
- Interpreter
- Compiler

TYPES OF DESIGN

Top-Down-Design

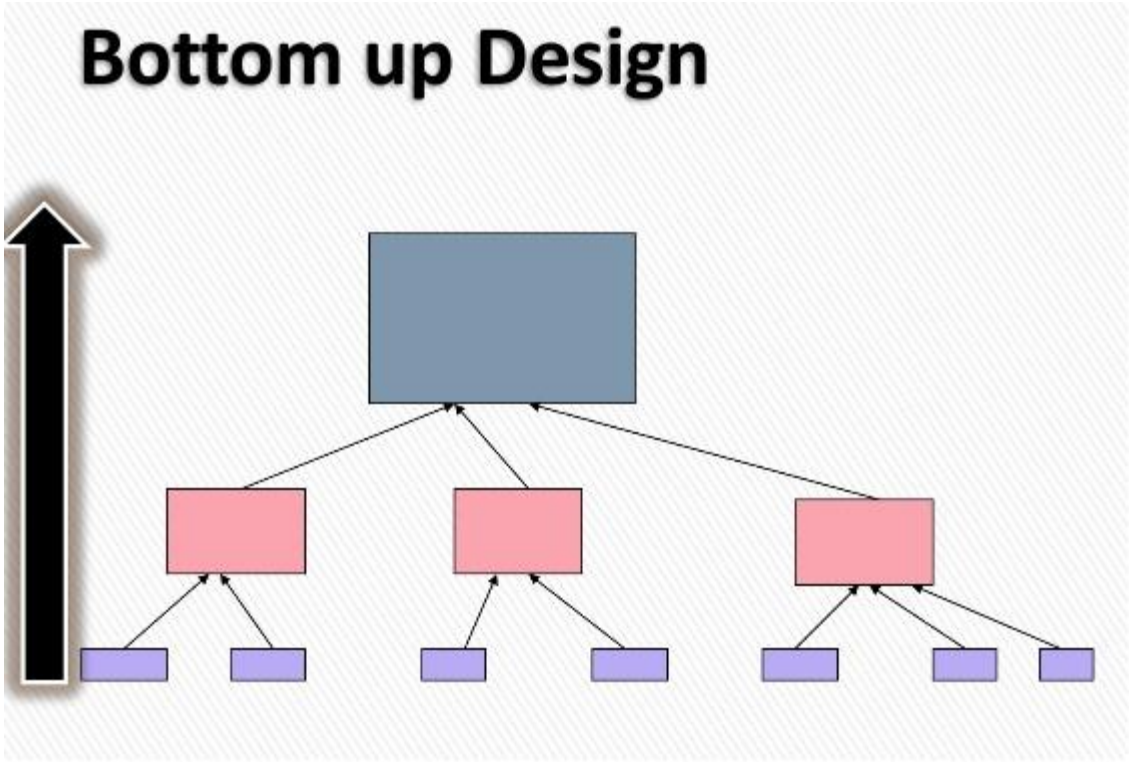
In top-down model, an overview of the system is formulated, without going into detail for any part of it.



TYPES OF DESIGN

Bottom-up Design

In bottom-up design individual parts of the system are specified in details.



STRUCTURE OF PROGRAM

- Comment section
- library section
- Declaration section
- Main()

```
{
    Variable declaration;
    Executable code;
}
```

EXAMPLE OF A PROGRAM

```
1   /* Fig. 2.1: fig02_01.c
2      A first program in C */
3   #include <stdio.h>
4
5   int main()
6   {
7      printf( "Welcome to C!\n" );
8
9      return 0;
10  }
```

Comments

- Text surrounded by /* and */ is ignored by computer

- #include <stdio.h>
  - Preprocessor directive

Example continue

- **`int main()`**
  - C++ programs contain one or more functions, exactly one of which must be **`main`**
  - Parenthesis used to indicate a function
  - **`int`** means that **`main`** "returns" an integer value
  - Braces (`{` and `}`) indicate a block
    - The bodies of all functions must be contained in braces

COMMENT

In computer programming a **comment** is a programmer-readable annotation in the source code of a computer program. They are added with the purpose of making the source code easier to understand, and are generally ignored by compilers and interpreters.

TYPES OF COMMENTS

- SINGLE-LINE COMMENTS
- MULTI-LINE COMMENTS

LIBRARY SECTION

A **library** is a collection of implementation of methods, written in terms of a language, that has a well-defined interface by which the method is invoked.

Method to include library functions in a program,

- #include <*.h>
- Imports *.class

DECLARATION SECTION

There are two types of declaration :-

Local Variables

These variables only exist inside the specific function that creates them. They are unknown to other functions and to the main program.

Global Variables

These variables can be accessed (i.e. known) by any function comprising the program. They are implemented by associating memory locations with variable names.

DECLARATION OF VARIABLES

```
int i=4; /* Global definition */
 main()
 {
 i++; /* global variable */
 func
 }
 func()
{
int i=10; /* Internal declaration */
 i++; /* Internal variable */
 }
```

SCOPE AND LIFETIME OF VARIABLE

Scope Variable	Lifetime Variables
The scope of a declaration is the part of the program for which the declaration is in effect.	The lifetime of a variable or object is the time period in which the variable/object has valid memory

LIFETIME VARIABLE EXAMPLE

A static variable is stored in the data segment of the "object file" of a program. Its lifetime is the entire duration of the program's execution. Example

```
#include <stdio.h>
Void func (){
Static int x=0;
X++;
Printf("%d\n",x);}
int main ()
func();
func();
Return 0;}
```

OTHER EXAMPLE

•**Automatic:** An automatic variable has a lifetime that begins when program execution enters the function or statement block or compound and ends when execution leaves the block. Automatic variables are stored in a "function call stack".

Dynamic: The lifetime of a dynamic object begins when memory is allocated for the object

•(e.g., by a call to malloc()) and ends when memory is deallocated (e.g., by a call to free()). Dynamic objects are stored in "the heap".

DATATYPES

INTRODUCTION

The data type of a value (or variable in some contexts) is an attribute that tells what kind of data that value can hold.

A data type is a classification identifying one of various types of data.

TYPES OF DATATYPES

Almost all programming languages explicitly include the notion of data type, though different languages may use different terminology. Common data types include

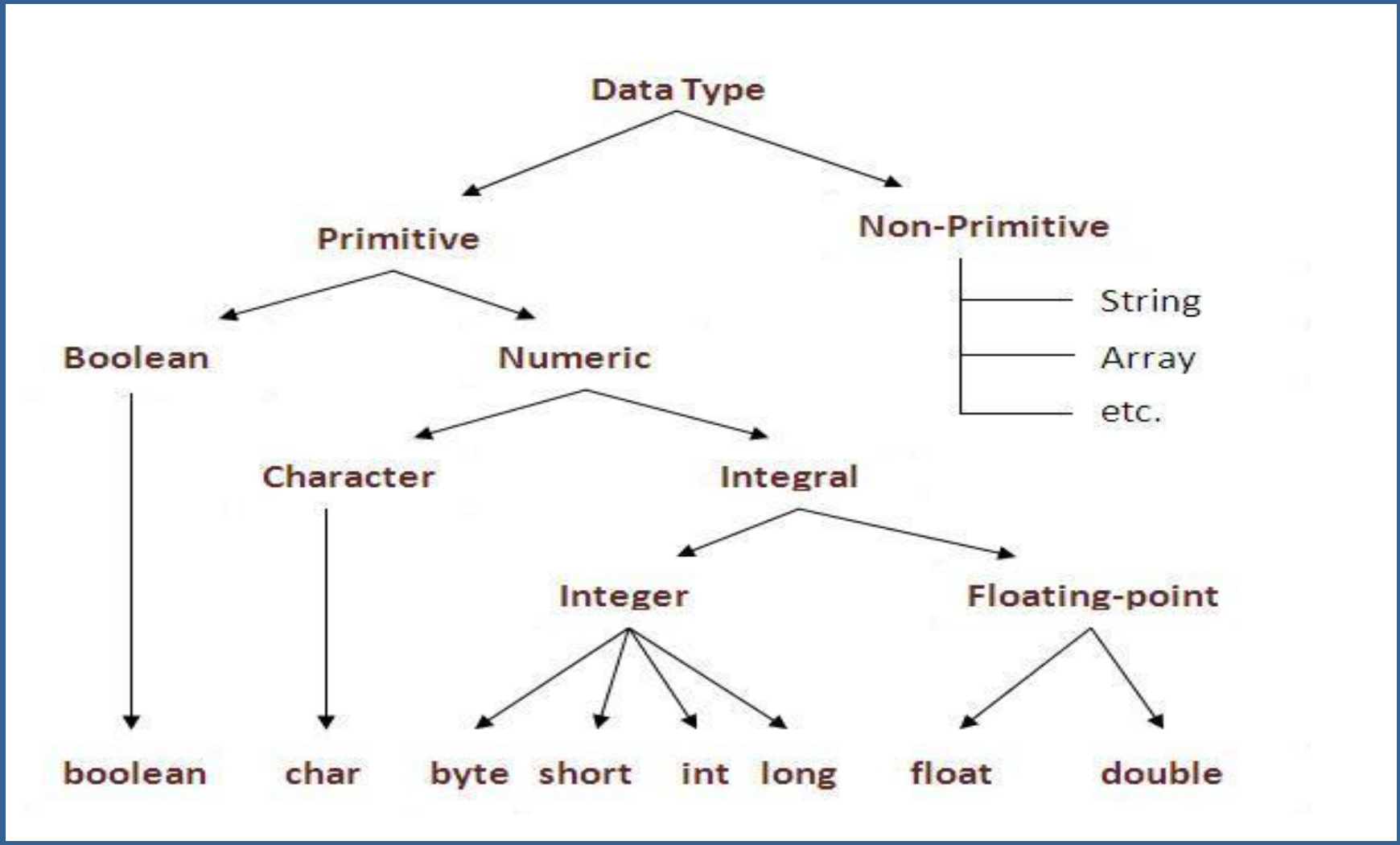
- Integers
- Booleans
- Characters
- Floating-point Numbers
- Alphanumeric Strings

ARRAY

Arrays, a kind of data structure that can store a fixed-size sequential collection of elements of the same type.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

DATATYPES IN JAVA



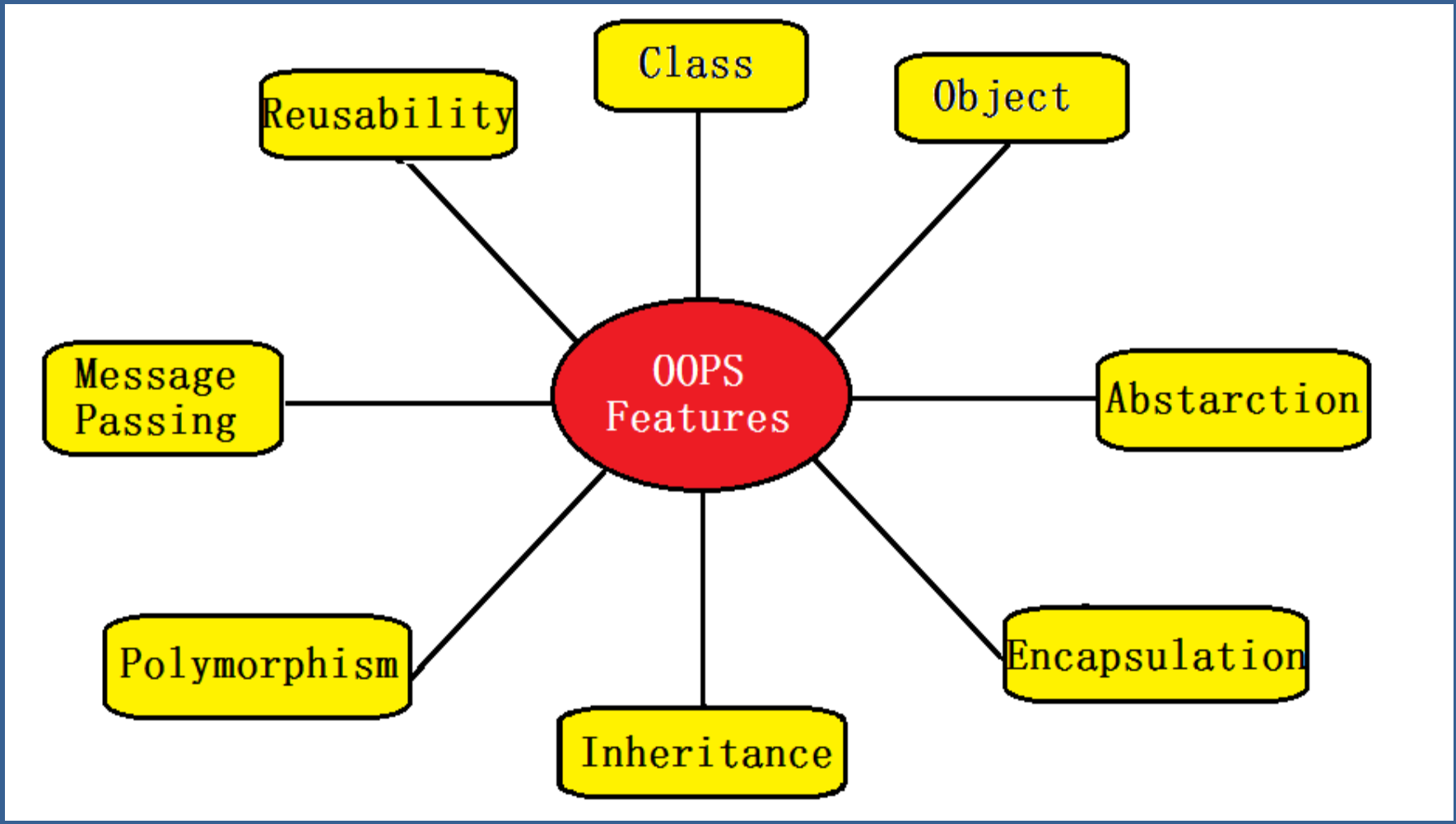
OBJECT ORIENTED PROGRAMMING SYSTEM

OOPS

Object-oriented programming (OOP) refers to a type of computer programming concept in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another.

CONCEPT OF OOPS



CONCEPTS

- Encapsulation – Encapsulation is capturing data and keeping it safely and securely from outside interfaces.
- Inheritance- This is the process by which a class can be derived from a base class with all features of base class and some of its own. This increases code reusability.
- Abstraction- The ability to represent data at a very conceptual level without any details.

OBJECT

Objects take up space in the memory and acts as instances of classes. When a program is executed , the objects interact by sending messages to one another. Each object contain data and code to manipulate the data. Objects can interact without having know details of each others data or code.

CLASS

Class is a collection of objects of similar type. Objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Eg: grapes and orange are the member of class fruit.

BINDING AND BINDING TIMES

A binding in a program is an association of an attribute with a program component such as an identifier or a symbol.

- Early Binding

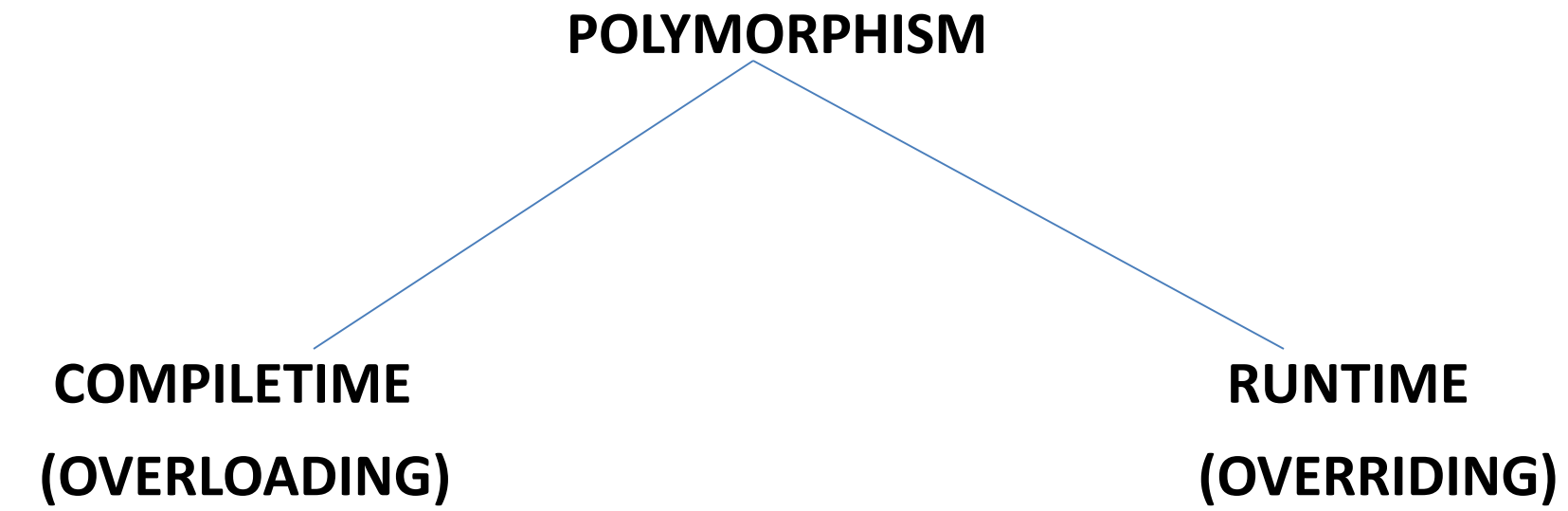
  method being called is looked up by name at COMPILE TIME.

- Late Binding

  method being called is looked up by name at RUNTIME.

POLYMORPHISM

The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism.



OVERLOADING

Overloading is ability of one function to perform different tasks i.e. it allows creating several methods with the same name which differ from each other in the type of the input and the output of the function.

Overloading can be used with functions and members.

OVERRIDING

Overriding is an object-oriented programming feature that enables a child class to provide different implementation for a method that is already defined or implemented in its parent class or one of its parent classes.

DATA STRUCTURE

INTRODUCTION

Data Structures provide a way to manage large amounts of data efficiently.

A data structure is a concept of organizing data in a computer so that it can be used efficiently and effectively.

LINKED LIST

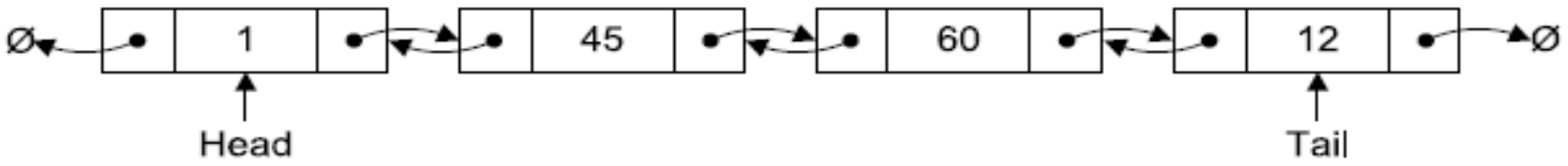
Linked List is a linear data structure and it is very common data structure which consists of group of nodes in a sequence which is divided in two parts.

Data

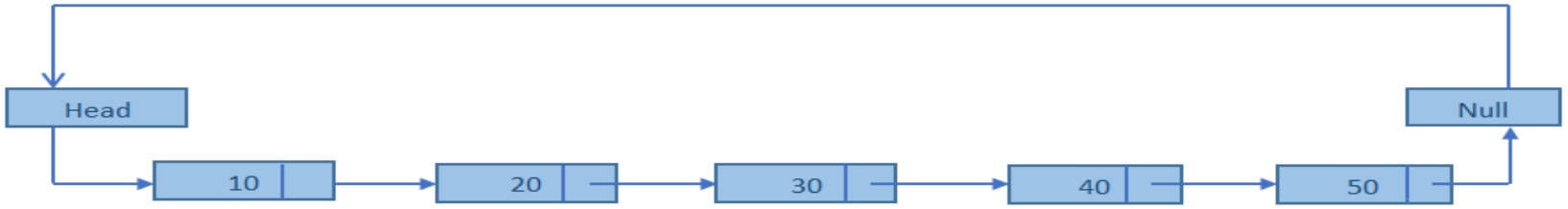
Address.

Linked Lists are used to create trees and graphs.

TYPES OF LINK LIST



DOUBLE LINK LIST

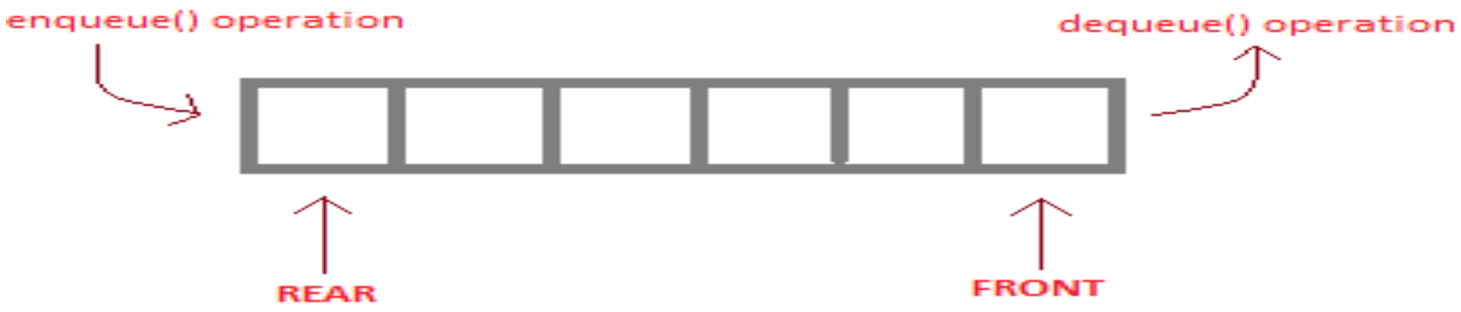


CIRCULAR LINK LIST

QUEUE

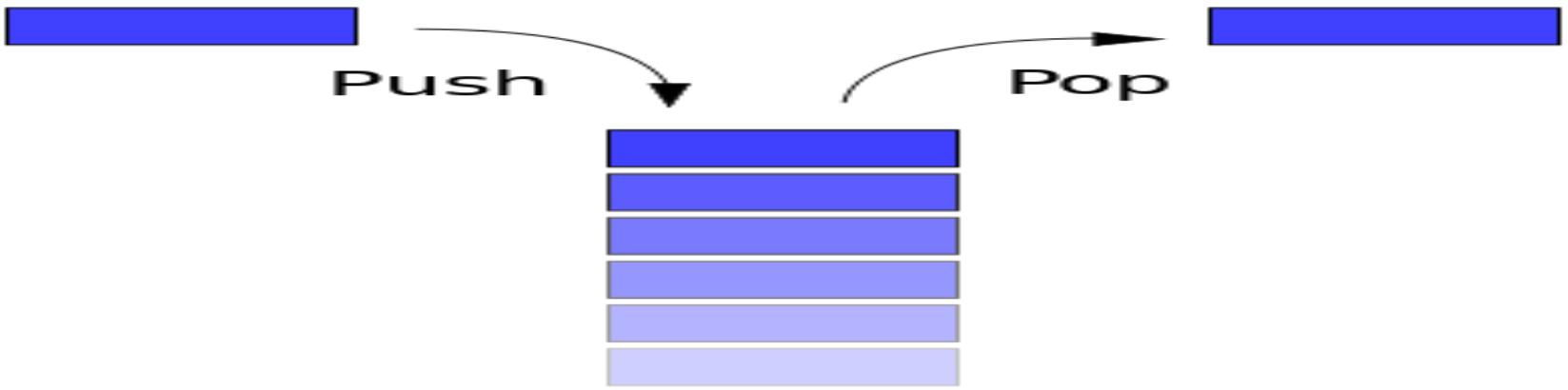
Queue is also an abstract data type or a linear data structure.

- element is inserted from one end called REAR
- Deletion takes place from the FRONT.
- This makes queue as FIFO data structure



STACK

- Stack is an abstract data type with a bounded(predefined) capacity.
- It operates on data by two operations
  - PUSH
  - POP
- Stack uses LIFO order



SORTING

Sorting is a technique of arranging data in a given manner. For example numbers can be arranged in ascending or descending manner.

	Worst Case	Average Case	Best Case
Bubble Sort	$O(n^2)$	$O(n^2)$	O(n)
Insertion Sort	$O(n^2)$	$O(n^2)$	O(n)
Merge Sort	O(n log n)	O(n log n)	O(n log n)
Heap Sort	O(n log n)	O(n log n)	O(n log n)
Quick Sort	$O(n^2)$	O(n log n)	O(n log n)


Thank You
shift